Efficient FPGA implementation of high-throughput mixed radix multipath delay commutator FFT processor for MIMO-OFDM

Dali, Mohammed; Guessoum, Abderezak; Gibson, Ryan M.; Amira, Abbes; Ramzan, Naeem

Published in:
Advances in Electrical and Computer Engineering

DOI:
10.4316/AECE.2017.01005

Publication date:
2017

Document Version
Author accepted manuscript

Efficient FPGA Implementation of High-Throughput Mixed Radix Multipath Delay Commutator FFT Processor for MIMO-OFDM

Mohammed DALI[1]*, Abderezak GUESSOUM[2], Ryan M. GIBSON[3], Abbes AMIRA[4,5], Naeem RAMZAN[4]

[1]*Research Laboratory in Electrical Engineering and Automatic, University of Medea,26000, Algeria*
[2]*Department of Electronics Engineering, University of Blida, 09000, Algeria*
[3]*School of Engineering and Built Environment, Glasgow Caledonian University, G4 0BA, UK*
[4]*School of Engineering and Computing, University of the West of Scotland, Paisley, PA1 2EB, UK*
[5]*Department of Computer Science and Engineering, Qatar University, Doha, 2713, Qatar*
**dali.mohamed @univ-medea.dz*

***Abstract*—This article presents and evaluates pipelined architecture designs for an improved high-frequency Fast Fourier Transform (FFT) processor implemented on Field Programmable Gate Arrays (FPGA) for Multiple Input Multiple Output Orthogonal Frequency Division Multiplexing (MIMO-OFDM). The architecture presented is a Mixed-Radix Multipath Delay Commutator. The presented parallel architecture utilizes fewer hardware resources compared to Radix-2 architecture, while maintaining simple control and butterfly structures inherent to Radix-2 implementations. The high-frequency design presented allows enhancing system throughput without requiring additional parallel data paths common in other current approaches, the presented design can process two and four independent data streams in parallel and is suitable for scaling to any power of two FFT size *N*. FPGA implementation of the architecture demonstrated significant resource efficiency and high-throughput in comparison to relevant current approaches within literature. The proposed architecture designs were realized with Xilinx System Generator (XSG) and evaluated on both Virtex-5 and Virtex-7 FPGA devices. Post place and route results demonstrated maximum frequency values over 400 MHz and 470 MHz for Virtex-5 and Virtex-7 FPGA devices respectively.**

***Index Terms*— Fast Fourier Transform (FFT), Field Programmable Gate Arrays (FPGA), MIMO, OFDM, Parallel architecture.**

I. INTRODUCTION

Multiple Input Multiple Output-Orthogonal Frequency Division Multiplexing (MIMO-OFDM) is considered as one of the most promising solution to achieve high-throughput in today's wireless communication systems. It is based on combining Multiple Input Multiple Output (MIMO) signal processing techniques [1] with Orthogonal Frequency Division Multiplexing (OFDM) modulation [2]. MIMO-OFDM significantly improves the reliability and throughput in wireless communication. Hence the technique is adopted with a wide range of current high-throughput communication standards, such as IEEE802.11, IEEE802.16, WiFi, WiMax, 3GPP and LTE [3-6]. The Fast Fourier Transform (FFT) and the Inverse Fast Fourier Transform (IFFT) processors are among the highest computationally complex and most critical modules within MIMO-OFDM based systems. They are required to process several independent high-throughput streams in parallel, while achieving real time requirements.

Literature indicates hardware design has migrated towards efficient and high performance FFT/IFFT architectures for MIMO-OFDM [7-14], and other applications, including signal and image processing [15-24]. Most of the existing architectures can be categorized into memory based designs, where only one butterfly is used, and pipeline designs, which uses a pipeline of butterfly stages. Pipeline architectures are suitable for MIMO-OFDM systems as they can achieve high-throughput and low-latency characteristics, which are required for continuous flow and real time applications [7], [11]. Pipeline architectures consist of two principal classes; Delay Feedback (DF) [15-18] and Delay Commutator [19-22]. The DF class includes Single-path Delay Feedback (SDF) and Multi-path Delay Feedback (MDF), where the DC class contains Single-path Delay Commutator (SDC) and Multi-path Delay Commutator (MDC).

MIMO-OFDM based systems require FFT/IFFT of parallel independent data sequences to be calculated simultaneously. The traditional approach is to use several FFT/IFFT processors, where each processor deals with one sequence. However, as the number of independent data sequences increase, the hardware complexity of the system becomes very high. Hence, proposing efficient FFT/IFFT architectures using a single processor to process several streams is crucial for MIMO-OFDM designs. In this context, Lin and Lee [7], Fu and Ampadu [8] presented a 64/128 point FFT/IFFT processor based on mixed radix decomposition for up to four independent data sequences. Yang, Tsai and Chuang [9] proposed a MDC based architecture and a memory scheduling method to obtain a variable length FFT/IFFT processor. The circuit was fabricated using UMC 90 nm CMOS technology and can process up to four independent sequences. Tang, Liao and Chang [10] developed a Flexible-Radix-Configuration Multipath-Delay-Feedback (FRCMDF) architecture for variable length and multiple stream-based FFT processor. The architecture was fabricated using a TSMC 18 µm CMOS process. Tang, Tsai and Chang [11] introduced a mixed radix MDF based architecture for 2048 points FFT with up to eight parallel data streams. The processor was

fabricated on 90 nm 1P9M CMOS and can obtain up to 2.4 GS/s throughput. Another mixed radix MDF based design for eight parallel data sequences was presented by Wang, Yan and Fu [12]. The design realizes a high-throughput and low-complexity 512-points FFT/IFFT processor. Additionally, Yan and Fu [12] proposed a single-Random Access Memory based group reorder that converts outputs into normal order. Tsai, Chen and Huang [13] proposed a systematic approach to automatically generate a variable size FFT/IFFT algorithm. Lin, Liu and Lee [14] proposed a Mixed Radix MDF architecture to implement a 128-point FFT/IFFT algorithm for Ultra Wide Band (UWB) communication systems.

FPGA FFT/IFFT algorithm implementations have been considered as a suitable solution to fulfil cost-efficient and high-performance implementations. Boopal, Garrido and Gustafsson [18] presented multi-streaming and variable-length Single-path Delay Feedback (SDF) based FFT architecture on Virtex-6 technology, which could process one stream of 2048-point, two streams of 1024-point or four streams of 512-point FFT operations. Ayinala, Brown and Parhi [20] proposed parallel MDC architectures using Radix-2, Radix-2^2 and Radix-2^3 independently to determine the FFT response of one real or complex sequence, which were synthesized on a Virtex-5 device. Garrido, Grajal, Sánchez and Gustafsson [21] presented a similar technique, which extended to include Radix-2^4 and any number of parallel paths which is a power of two, with implementation on Virtex-5. Further work from Garrido, Acevedo, Ehliar and Gustafsson [22] investigated FFT performance limitations carried out on Virtex-6. Wang, Liu, He, and Yu [23] proposed a Single-path Delay Commutator-Feedback (SDC-SDF) combination technique based on Radix-2 and synthesized on Virtex-5 devices. Uzun, Amira and Bouridane [24] presented a common framework for a memory-based FFT algorithm implemented on FPGA. This method is suitable for extension to include image processing applications.

Radix-2^k MDC architecture has been more recently presented as an efficient solution for implementing FFT/IFFT algorithms [20-22]. Radix-2^k MDC architecture can deal with any number of parallel data samples which are a power of two and can achieve high-throughput performance. In addition, the architecture requires fewer hardware operators compared to parallel feedback architecture MDF [21]. However, existing Radix-2^k MDC designs are limited to processing single data streams in parallel and cannot process multiple independent data streams as required in MIMO-OFDM systems. Additionally, architectures based on single Radix-2^k limits the design to processing sequences of data in correspondence with power of 2^k sizes only.

This paper presents a high-frequency Mixed Radix-2^2 Radix-2^3 Multipath Delay Commutator (MR2^2-2^3 MDC) FFT processor for MIMO-OFDM systems, with two and four data streams. The proposed design is suitable for implementing any power of two FFT size, where the design offers significant FPGA resource-space savings and significant utilization efficiency of embedded multipliers. Furthermore, the mixed radix approach allows FFT processor of size N to be constructed with various Radix-2^2 and Radix-2^3 module combinations. The remaining sections of this paper are organized as follows: Section II explains the Radix-2^2 and Radix-2^3 FFT algorithms. Section III introduces the proposed high-frequency MR2^2-2^3 MDC. Section IV explains the design flow and implementation of the proposed architecture on FPGA, while Section V presents the experimental results of the implemented architecture on FPGA. Conclusions are provided in Section VI.

II. Algorithmic Review

A. Radix 2^2 Algorithm

The N-point Discrete Fourier Transform (DFT) is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n).W_N^{nk} \tag{1}$$

where $x(n)$ and $X(k)$ are complex samples in time and frequency domains, while the twiddle factor is defined as $W_N^{nk} = \exp(\frac{-j2\pi nk}{N})$

Consider the following three-dimensional mapping of n and k [17]:

$$n = < \frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3 >_N \tag{2}$$

$$k = < k_1 + 2k_2 + 4k_3 >_N \tag{3}$$

where the notation $< >_N$ represents the *modulo-N* operator.

Substituting equations (2) and (3) into equation (1) obtains:

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x(\tfrac{N}{2} n_1 + \tfrac{N}{4} n_2 + n_3) W_N^{(\frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3)(k_1 + 2k_2 + 4k_3)} \tag{4}$$

Summing among n_1 and considering:

$$B_{\frac{N}{2}}(\tfrac{N}{4} n_2 + n_3, k_1) = x(\tfrac{N}{4} n_2 + n_3) + (-1)^{k_1} x(\tfrac{N}{4} n_2 + n_3 + \tfrac{N}{2}) \tag{5}$$

Equation (4) can be re-formulated as follows:

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} \left[B_{\frac{N}{2}}(\tfrac{N}{4} n_2 + n_3, k_1) W_N^{(\frac{N}{4} n_2 + n_3) k_1} \right] W_N^{(\frac{N}{4} n_2 + n_3)(2k_2 + 4k_3)} \tag{6}$$

The critical concept of Radix-2^2 is to cascade the twiddle factor $W_N^{(\frac{N}{4} n_2 + n_3) k_1}$ into the next step decomposition instead of multiplying with $B_{\frac{N}{2}}(\frac{N}{4} n_2 + n_3, k_1)$:

$$W_N^{(\frac{N}{4} n_2 + n_3) k_1} W_N^{(\frac{N}{4} n_2 + n_3)(2k_2 + 4k_3)} = (-j)^{n_2 (k_1 + 2k_2)} W_N^{n_3 (k_1 + 2k_2)} W_N^{4 n_3 k_3} \tag{7}$$

Substituting (7) into (6) and summing among n_2 determine:

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \left[H_{\frac{N}{4}}(n_3, k_1, k_2) W_N^{n_3 (k_1 + 2k_2)} \right] W_{\frac{N}{4}}^{n_3 k_3} \tag{8}$$

where $H_{\frac{N}{4}}(n_3, k_1, k_2)$ can be expressed as follows:

$$H_{\frac{N}{4}}(n_3, k_1, k_2) = B_{\frac{N}{2}}(n_3, k_1) + (-j)^{(k_1 + 2k_2)} B_{\frac{N}{2}}(n_3 + \tfrac{N}{4}, k_1) \tag{9}$$

Equation (5) represents the first butterfly structure containing only addition and subtraction operations. Equation (9) demonstrates the second butterfly structure containing addition, subtraction and multiplication with –j.

The Radix-2^2 algorithm requires full multiplication every two stages as demonstrated in equation (8), opposed to multiplication occurring during every stage within Radix-2. Hence, Radix-2^2 demonstrates a 50% reduction of multiplication operations required in comparison to Radix-2. The complete Radix-2^2 algorithm can be obtained through decomposing equation (8) recursively to the remaining $\frac{N}{4}$ points DFTs.

The fundamental structure to obtain two consecutive stages in the Signal Flow Graph (SFG) for Radix-2^2 Decimation In Frequency (DIF) algorithm is illustrated in Fig. 1, where $x_i(.)$ denotes the inputs of stage i and $X_{i+1}(.)$ denotes the outputs of stage $i+1$. When n is varied from 0 to $Ni/4-1$ in the basic structure of Fig. 1 a group of N_i inputs is formed , where N_i is given by:

$$N_i = \frac{N}{2^{(i-1)}} = 2^{n_s - i + 1} \tag{10}$$

The group is repeated N/N_i times to obtain two consecutive stages in the complete SFG.

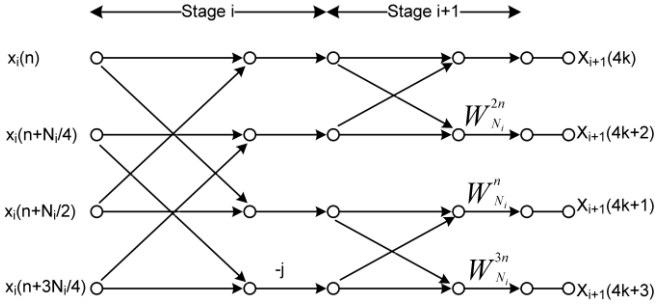


Figure 1. SFG of the basic butterfly structure of Radix-2^2 DIF algorithm

B. *Radix 2^3 Algorithm*

Applying four dimensional linear index mapping to n and k can be expressed as follows:

$$n = < \frac{N}{2} n_1 + \frac{N}{4} n_2 + \frac{N}{8} n_3 + n_4 >_N \tag{11}$$

$$k = < k_1 + 2k_2 + 4k_3 + 8k_4 >_N \tag{12}$$

The DFT function in equation (1) can be expressed in terms of four-dimensional linear index mapping, as follows:

$$X(k_1 + 2k_2 + 4k_3 + 8k_4) = \sum_{n_4=0}^{\frac{N}{8}-1} \sum_{n_3=0}^{1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x(\tfrac{N}{2} n_1 + \tfrac{N}{4} n_2 + \tfrac{N}{8} n_3 + n_4) W_N^{nk} \tag{13}$$

Additionally, the twiddle factor W_N^{nk} with cascade decomposition can be expressed as:

$$\begin{aligned} W_N^{nk} &= W_N^{\frac{N}{2} n_1 k_1} W_N^{\frac{N}{4} n_2 (k_1 + 2k_2)} W_N^{\frac{N}{8} n_3 (k_1 + 2k_2 + 4k_3)} W_N^{n_4 (k_1 + 2k_2 + 4k_3 + 8k_4)} \\ &= (-1)^{n_1 k_1} (-j)^{n_2 (k_1 + 2k_2)} W_N^{\frac{N}{8} n_3 (k_1 + 2k_2 + 4k_3)} W_N^{n_4 (k_1 + 2k_2 + 4k_3)} W_{\frac{N}{8}}^{n_4 k_4} \end{aligned} \tag{14}$$

Substitution of equation (14) into equation (13) and expending summation among n_1, n_2 and n_3 determine:

$$X(k_1 + 2k_2 + 4k_3 + 8k_4) = \sum_{n_4=0}^{\frac{N}{8}-1} \left[T_{\frac{N}{8}}(n_4, k_1, k_2, k_3) W_N^{n_4 (k_1 + 2k_2 + 4k_3)} \right] W_{\frac{N}{8}}^{n_4 k_4} \tag{15}$$

Eight DFT functions of length $\frac{N}{8}$ were expressed in equation (15), where a third butterfly structure $T_{\frac{N}{8}}(n_4, k_1, k_2, k_3)$ is expressed as follows:

$$\begin{aligned} &T_{\frac{N}{8}}(n_4, k_1, k_2, k_3) = \\ &H_{\frac{N}{4}}(n_4, k_1, k_2) + W_N^{\frac{N}{8}(k_1 + 2k_2 + 4k_3)} H_{\frac{N}{4}}(n_4 + \tfrac{N}{8}, k_1, k_2) = \\ &H_{\frac{N}{4}}(n_4, k_1, k_2) + (\tfrac{\sqrt{2}}{2}(1 - j))^{k_1} (-j)^{(k_2 + 2k_3)} H_{\frac{N}{4}}(n_4 + \tfrac{N}{8}, k_1, k_2) \end{aligned} \tag{16}$$

The third butterfly structure involves the twiddle factor $(\frac{\sqrt{2}}{2}(1-j))^{k_1}(-j)^{(k_2+2k_3)}$, which can be realized with constant scalar operations. The initial radix-2^3 SFG stage contains only trivial multiplication with $-j$, the second stage multiplies with $-j$ and constants W_8^1 and W_8^3, while the third stage contains multiplications with the general term. $W_N^{n_4(k_1+2k_2+4k_3)}$. In comparison, Radix-2 required multiplication during every stage, while Radix-2^2 only uses multiplication operations every second stage.

The fundamental structure to obtain three consecutive stages in the SFG of Radix-2^3 DIF algorithm is presented in Fig. 2. A group of N_i inputs is obtained when n varies from 0 to $N_i/8-1$. Three consecutive stages in the complete SFG are formed by N/N_i groups.

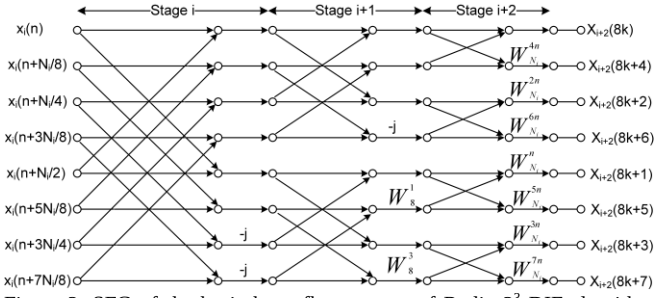


Figure 2. SFG of the basic butterfly structure of Radix-2^3 DIF algorithm

III. Proposed High-Frequency MR2^2-2^3 MDC Architecture

To the best of the authors' knowledge, current literature has not investigated improved high-frequency MDC architecture implementations on FPGA through added delay elements. Current architectures within literature have significantly limited maximum clock frequency responses when synthesized on FPGA without utilizing pipeline registers at element outputs. This poor maximum frequency performance occurs frequently when FFT size increases [19-22]. Using an appropriate number of pipeline registers at the output element within the FFT architecture allows the length of critical paths within the processor circuit to be significantly reduced and achieve high-frequency response. The latency introduced with pipeline registers at each element output can be defined as the number of clock sample periods used for the block output delay. Simulation-based investigations indicate the appropriate number of pipeline registers to achieve maximum frequency is: four for complex multipliers, one for addition, subtraction and multiplexer and two for Block Random Access Memory (BRAM) operations. Pipeline register associated latencies cause synchronization issues within the architecture, affecting the FFT function response. Hence, it is critical to compensate the effect of pipeline register latency through inserting additional delay elements at specific points in the architecture design.

The general structure for the proposed MR-2^2-2^3 MDC architecture is demonstrated in Fig. 3. The architecture is

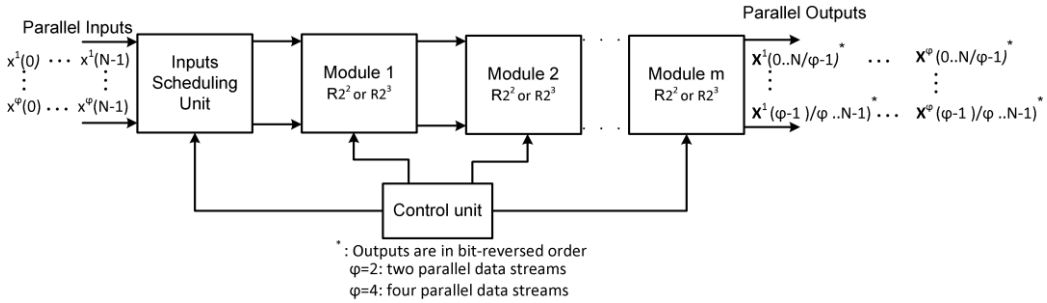


Figure 3. General structure of the proposed MR2^2-R2^3 MDC architecture

composed of an input memory-scheduling unit, *m* modules, where each module consists of either a Radix-2^2 or Radix-2^3 butterfly structures with a control unit. Let m_2 denote the number of Radix-2^2 modules, m_3 the number of Radix-2^3 modules and n_s the number of stages in the Signal Flow Graph (SFG), ($n_s = \log_2(N)$), then n_s, m_2, and m_3 are related as follows:

$$n_s = 2m_2 + 3m_3 \quad (17)$$

Equation (17) demonstrates that utilizing MR-2^2-2^3 allows scaling FFT size to any power of two, which is not possible to achieve with single Radix-2^2 or Radix-2^3.

The Radix-2^2 function performs operations in two stages within the SFG as detailed in Fig. 1. In contrast, the Radix-2^3 function performs operations in three stages within the SFG as detailed in Fig. 2. For example, a FFT size *N=32* demonstrates a SFG with five stages, which can be realized within architecture requiring one Radix-2^2 operation pipelined with one other Radix-2^3 operation.

A. Proposed Architecture for Two Data Streams

The proposed FFT system determines the DFT responses for two parallel data streams; *A* and *B*. The input memory scheduling system arranges the input sequences of data streams *A* and *B* as shown in Fig. 4. Data stream $A = x^1(0..N-1)$ is separated into two sub-streams, obtaining upper and lower sub-streams $A_1 = x^1(0..\frac{N}{2}-1)$ and $A_2 = x^1(\frac{N}{2}..N-1)$, where sub-stream length is *N/2*. Similarly, data stream *B* is separated into upper sub-stream B_1 and lower sub-stream B_2. The input data stream sequences are reshuffled to allow sub-streams A_1 and A_2 to be processed during the initial time index with Module 1 through *m* as shown in Fig. 3, followed by processing sub-steams B_1 and B_2. Modules 1 through *m* can consist of Radix-2^2 or Radix-2^3 structures as shown in Fig. 5 and Fig. 6. Pipeline register latencies associated for operation functions have been presented within block diagrams. Delay elements are operated to maintain data synchronization with upper and lower sub-streams. Operations are applied to the upper U_{in} and lower L_{in} input data sub-streams to achieve the upper U_{out} and lower L_{out} output data sub-streams.

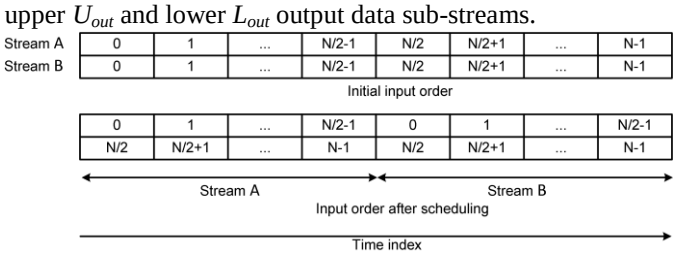


Figure 4. Input reordering with Input Scheduling Unit for two parallel streams.

The Radix-2^2 MDC module includes two addition/substruction functions, one trivial multiplier, two standard multipliers, two Read Only Memories (ROM), two shuffling units, two multiplexers and three delay elements. The trivial multiplier performs real-imaginary swap with sign inversion operations.

Shuffling units are inserted between each consecutive stages, i.e. a shuffling unit is inserted between stage *i* and stage *i+1*, as shown in Fig. 5. They are built using *2D* delay elements, where $D = N/2^{(i+1)}$ and requires two multiplexers. The delay elements are used for synchronizing the previous stage data outputs, *i.e.* stage *i*, where multiplexers allow routing data to the corresponding input of the Add/Sub unit of the stage *i+1*. The multiplexer control signal C_{n_s-i-1} is directly obtained from a binary counter. It takes the value zero for *D* clock cycles followed by the value one for *D* others clock cycles, *i.e.* $C_{n_s-i-1} = \overbrace{00..0}^{D}\overbrace{11..1}^{D}$. For *D* clock cycles, the upper data outputs of stage *i* U_1 to U_D are routed to the upper input of stage *i+1*, while simultaneously data outputs U_{D+1} to U_{2D} are routed to the lower input of stage *i+1*. The process is similarly repeated with the lower data outputs of stage *i* during the next *D* clock cycles, *i.e.* lower data L_1 to L_D are routed to the next stage upper input, while data L_{D+1} to L_{2D} are routed to the next stage lower input. Twiddle factors $W_{N_i}^{2n}$ with *n=0..N$_i$/4-1* are stored in the upper path's ROM (*ROM U)* of size $N_i/4$ while twiddle factors $W_{N_i}^{n}$ and $W_{N_i}^{3n}$ are stored in the lower path's ROM (*ROM L)* of size $N_i/2$. *ROM U* is addressed using bits C_0 to C_{n_s-i-2} of a binary counter while *ROM L* is addressed using bits C_0 to C_{n_s-i-1} of the same counter.

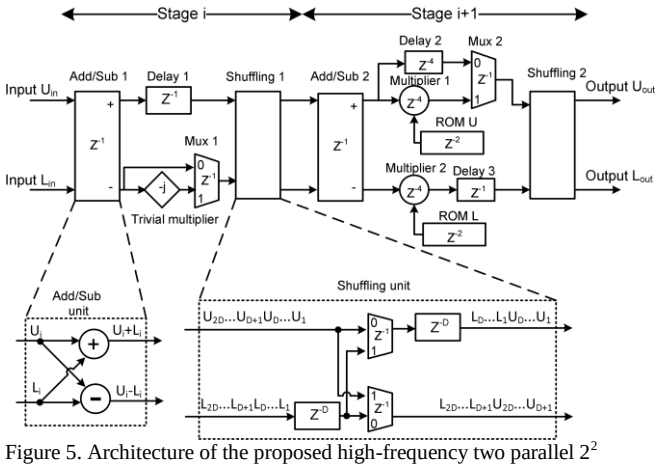


Figure 5. Architecture of the proposed high-frequency two parallel 2^2 module.

The Radix-2^3 module utilizes three addition/substraction functions, one trivial multiplier, two standard multipliers, one Constant Complex Multiplication Unit (CCMU1), two ROMs, three shuffling units, one multiplexer and two delay elements. The CCMU1 operation applies rotation with $-j$, W_8^1 and W_8^3, where the associated structure is presented in Fig. 6. The CCMU1 contains one complex scaling operator, two trivial multipliers, three multiplexers in addition to delay elements. The same complex scaling operator used to rotate with W_8^1, is cascaded with a trivial multiplier *-J* to accomplish rotation with W_8^3. Twiddle factors $W_{N_i}^0$, $W_{N_i}^{4n}$, $W_{N_i}^{2n}$ and $W_{N_i}^{6n}$ with *n=0..Ni/8-1* are stored in *ROM U*, while twiddle factors $W_{N_i}^{n}$, $W_{N_i}^{5n}$, $W_{N_i}^{3n}$ and $W_{N_i}^{7n}$ are stored in *ROM L*. Both *ROM U* and *ROM L* are of size $N_i/2$ and are addressed using bits C_0 to C_{n_s-i-1} of a binary counter. It is important to note that all operations after the last addition unit are removed in both Radix-2^2 and Radix-2^3 functions, when they are implemented within the architecture end stage of module *m*.

B. Proposed Architecture for Four Data Streams

Input scheduling of four independent data streams *A*, *B*, *C* and *D* can be obtained using the input memory scheduling technique proposed by Yang, Tsai and Chuang [9], which rearranges the input data as shown in Fig. 7. Each of the parallel data streams *A*, *B*, *C* and *D* are separated into four equal length sub-streams adhering to the following format: Data stream $A = x^1(0..N-1)$ is separated into $A_1 = x^1(0..\frac{N}{4}-1)$, $A_2 = x^1(\frac{N}{4}..\frac{N}{2}-1)$, $A_3 = x^1(\frac{N}{2}..\frac{3N}{4}-1)$ and $A_4 = x^1(\frac{3N}{4}..N-1)$, where sub-stream lengths are defined as *N*/4. Streams *B*, *C* and *D* are split into four equal length sub-streams in a similar manner.

Once the input data streams have been rescheduled with the memory scheduling unit, then Modules 1 through *m* determine the DFT response of streams A, *B*, *C* and *D* consecutively.

The four parallel Radix-2^2 and Radix-2^3 MDC module process four samples in a continuous flow. Architecture of an initial stage four parallel Radix-2^2 module is shown in Fig. 8, while the architecture for intermediate and last modules are presented in Fig. 9.

Shuffling units keep the same structure shown in Fig. 5, with *D* delays element for each unit instead of *2D* delays used within the two parallel data stream architecture. Twiddle factors $W_{N_i}^{2n}$, $W_{N_i}^{n}$, and $W_{N_i}^{3n}$ with *n=0..Ni/4-1* are stored in *ROM L1*, *ROM U2* and *ROM L2* respectively. The three memories each of size $N_i/4$ are addressed by bits C_0 to C_{n_s-i-2} of a binary counter.

Architecture design for the initial four-parallel Radix-2^3 module is illustrated in Fig. 10, where the structure for intermediate and last modules are shown in Fig. 11. Twiddle factors $(W_{N_i}^0, W_{N_i}^n)$, $(W_{N_i}^{4n}, W_{N_i}^{5n})$, $(W_{N_i}^{2n}, W_{N_i}^{3n})$ and $(W_{N_i}^{6n}, W_{N_i}^{7n})$ with *n=0..Ni/8-1* are stored in memories *ROM U1*, *ROM L1*, *ROM U2* and *ROM L2* respectively. The four memories, each of size $N_i/4$ are addressed by bits C_0 to C_{n_s-i-2} of a binary counter.

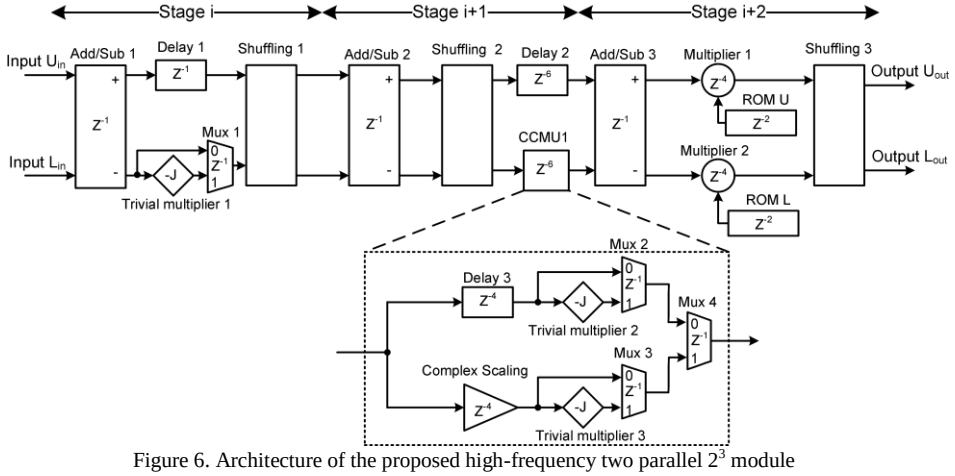


Figure 6. Architecture of the proposed high-frequency two parallel 2^3 module

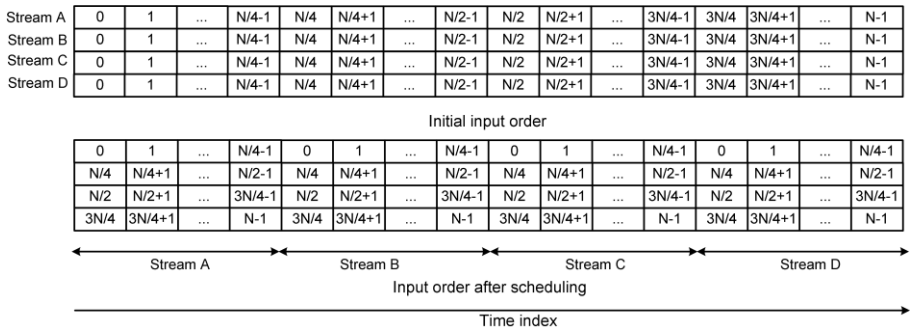


Figure 7. Input reordering by Scheduling Unit for four parallel data streams

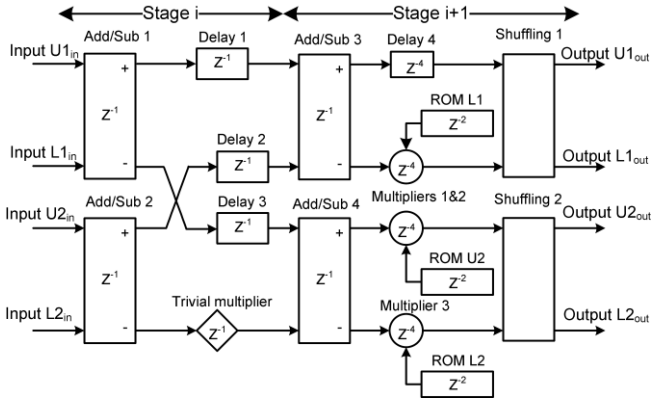


Figure 8. Architecture of the proposed high-frequency four parallel Radix-2^2 module as first processing module (*i=1*)

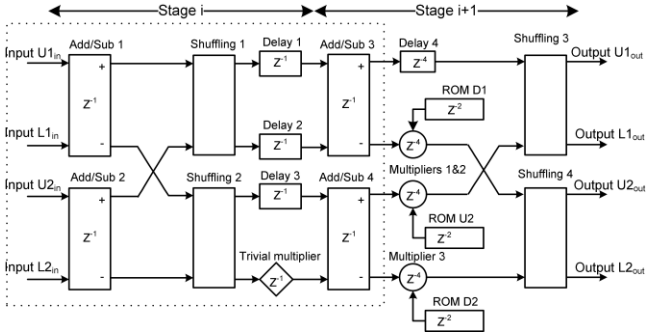


Figure 9. Architecture of the proposed high-frequency four parallel Radix-2^2 module as intermediate module (whole diagram) or as last module (only components within dotted section)

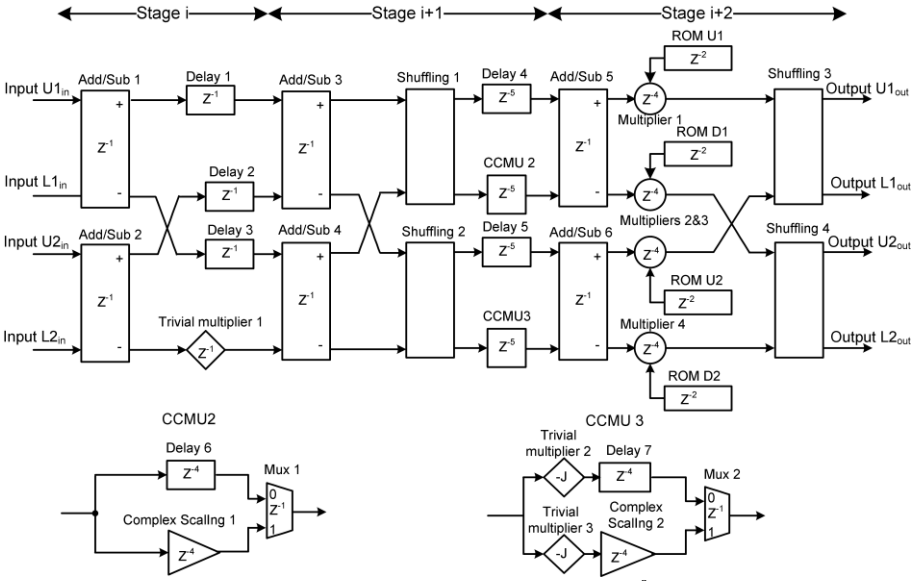


Figure 10. Architecture of the proposed high-frequency four parallel Radix-2^3 module as first module (*i=1*)

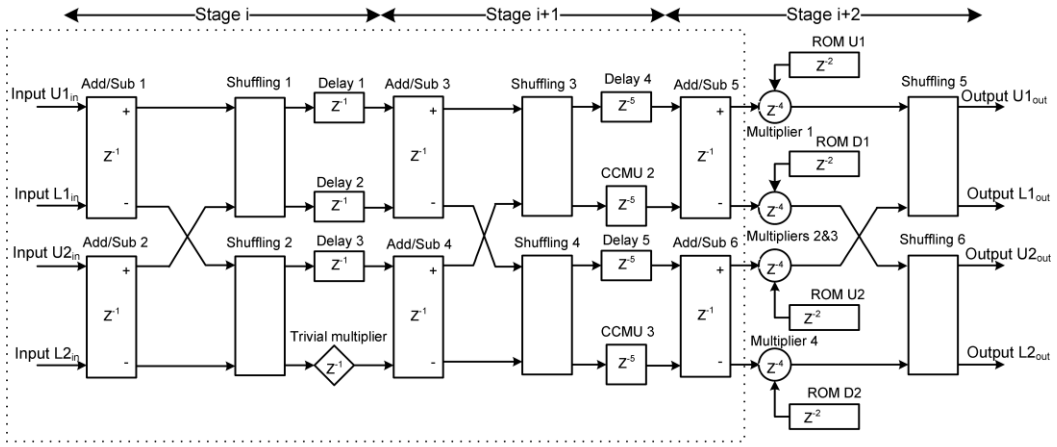


Figure 11. Architecture of the proposed high-frequency four parallel Radix-2^3 module as intermediate module (whole diagram) or as last module (only diagram within dotted rectangle).

It is important to note that both Radix-2^2 and Radix-2^3 initial and last modules (Module 1 and Module *m*) implement slightly modified architecture as presented in Fig. 8 to Fig. 11. Module 1 removes the shuffling operation at the output of the modules initial addition function, while Module *m* removes all operations after the last addition units.

C. Hardware Architecture Resource Requirements

The hardware architecture requirements for the proposed systems are presented and compared with other relevant work in Table I, where φ represents the number of independent data streams and *p* represents the number of parallel paths. The number of complex addition operators is dependent on the number of parallel data streams and FFT size *N* for any combination of Radix-2^2 and Radix-2^3 modules, where four data streams require double the amount of complex addition operators than a two data streams system. The memory architecture requirements presented in Table I are obtained with input scheduling unit, in addition to shuffling units inserted in Module 1 through *m*.

TABLE I. THE HARDWARE COMPLEXITY OF THE PROPOSED HIGH-FREQUENCY MR2^2-2^3 MDC ARCHITECTURE IN COMPARISON WITH OTHER ARCHITECTURES

Designs	Type	Radix	N	φ	P	Complex Adders	Memory size	Complex Multipliers	Rotation by W_8
Proposed	MDC	MR-2^2-2^3	2^n	2	2	$4\log_4 N$	$2N+2+(11m_3+6m_2-5k_1)^*$	2(m-1)	m_3
				4	4	$8\log_4 N$	$4N+2+(21m_3+7m_2-4k_1)^*$	$3m_2+4m_3+k_1-4$	$2m_3$
Lin *et al.* [7]	SDF	R-2 /R-8	64, 128	4	4	48	508	2+(4×0.62)	0
Fu *et al.* [8]	SDF/MDC	R2 /R4	64, 128	4	4	32	508	4	5
Yang *et al.* [9]	MDC	R4/ R8	128, 512, 1024, 2048	4	4	$8\log_4 N$	$3N+\sum_{s=1}^{\lfloor \log_4 N \rfloor -1} \frac{3N}{4^s}$	$3(\log_4 N-1)$	0
Garrido *et al.* [21], [22]	MDC	$R2^2$	4^n	1	2	$4\log_4 N$	N	$2(\log_4 N-1)$	0
					4	$8\log_4 N$	N	$3(\log_4 N-1)$	0
		$R2^3$	8^n	1	2	$4\log_4 N$	N	$3(\log_8 N-1)$	$\log_8 N$
					4	$8\log_4 N$	N	$4(\log_8 N-1)$	$2\log_8 N$

()* presents additional delay elements for high throughput design

For the proposed architectures, the additional delay elements used for high-frequency design are also considered

In two data streams architecture, the input scheduling unit uses two memory operators each of size *N/2*, with four additional delays to maintain data synchronization, where the shuffling units contain delay elements of total size *N-2*. Radix-2^3 requires 11 additional delays regardless of module position within the architecture. Radix-2^2 implements six additional delays for module rank positions 1 through *m-1* and one delay for module position rank *m*. Hence, the total memory size in the two parallel architecture is $2N+2+6m_2+11m_3-5k_1$, where $k_1=1$ if module *m* is a Radix-2^2, else $k_1=0$.

Systems with four parallel processing architectures involve an input scheduling unit with 12 memory banks, each of size *N/4* in addition to six further delay elements for data synchronization, producing a total memory size of *3N+6*. The shuffling units utilize delay elements of total size *N-4*. Each Radix-2^3 module obtains 21 additional delays independently of rank position. Additionally, Radix-2^2 requires seven additional delays for module rank position 1 through *m-1*, where module in rank position *m* uses three delays. Thus, the total memory size in the four parallel architecture is $4N+2+21m_3+7m_2-4k_1$.

Each module from rank position 1 through *m-1* implemented within the two parallel data stream architecture systems contain two complex multipliers, determining the total multipliers used as *2(m-1)*. Furthermore, each Radix-2^3 module contains a single rotation by W_8, carried out using the CCMU1 unit as shown in fig. 6. As a result, the total number of rotations by W_8 within the two parallel architecture is equal to m_3.

For systems with four parallel data stream architecture, Radix-2^2 modules located in rank position 1 through *m-1*, contain three complex multipliers, while Radix-2^3 modules in comparison require four multipliers at the same rank position. Hence, the total number of complex multipliers required in the four parallel architecture is $3(m_2-k_1)+4(m_3+k_1-1)$. Moreover, each Radix-2^3 module within the four parallel architecture contains two rotations by W_8 realized with CCMU2 and CCMU3 as illustrated in Fig. 10 and Fig. 11. Hence the total number of rotators by W_8 within the four parallel architecture is $2m_3$.

The proposed hardware architecture requirements are presented and compared with other relevant work in Table I. The proposed architecture is suitable for parallel data stream processing with FFT size $N = 2^{n_s}$, where other architectures evaluated have not taken these characteristics into design consideration.

Pipelined Radix-2^k architectures presented in [21] and [22] are capable of processing one data stream with parallel data paths, requiring additional input modules to process parallel data streams. Table I illustrates that the proposed architecture involves the same number of required complex addition operators as architecture systems in [9, 21, 22] and fewer than [7, 8]. Furthermore, some Radix-2^2-2^3 proposed combinations utilize fewer complex multipliers than equivalent architectures in [9, 21, 22]. For example, a FFT size *N=256* would require 6 and 9 multipliers for two and four parallel data streams in [9, 21, 22], while the proposed 2^3-2^2-2^3 combination uses 4 and 7 multipliers for two and four parallel data stream architecture. The proposed architecture combination obtains 33.3% and 22.2% reduction in required complex multipliers than architectures [9, 21, 22]. The FFT processor presented by Lin and Lee [7] requires two complex multipliers in addition to four modified complex multipliers which realize 0.62% of the standard multiplier area; this structure uses an equivalent of 4.48 multipliers in comparison to 6 multipliers used in this article presented method for the combination 2^2-2^2-2^3. However, work presented in [7] is limited to FFT size *N=64* and *N=128*, while the presented architecture is scalable to any power of two FFT size. Furthermore, architecture presented by Fu and Ampadu [8] requires fewer multipliers than the proposed system, however the architecture is also limited to FFT size *N=64* and *N=128*. The additional memory usage is limited within the proposed architecture to achieve high-frequency in the implementation.

IV. DESIGN FLOW AND IMPLEMENTATION

The proposed MR2^2-2^3 MDC architectures for various FFT size *N* are designed using Xilinx System Generator (XSG). The design is based on fixed-point number representation, where scaling operations are introduced to avoid numerical saturation. Scaling is performed with bit shifting to the right of processed data within early stages of the FFT process. XSG integration with the Matlab-Simulink environment allowed FFT input generation, in addition to evaluating the obtained FFT output. For verification, the output from the proposed fixed-point FFT design is compared to the output from Matlab's floating-point FFT function. Designs from XSG are utilized for generating Register Transfer Level (RTL) files for migration to Xilinx ISE design suite environment for FPGA implementation.

The architecture designs were synthesized and evaluated on a Virtex-5 FPGA XC5VSX240T-2FF1738 and Virtex-7 FPGA XC7VS485T-2FFG1661.

V. EXPERIMENTAL RESULTS

The proposed architecture was analyzed and evaluated with place and route synthesis. Mixed Radix-based methods allow FFT processors to be realized with various combinations of Radix-2^2 and Radix-2^3. For example an FFT size *N=256* can be built with four consecutive Radix-2^2 modules (2^2-2^2-2^2-2^2) or with a consecutive sequence of one Radix-2^3, one Radix-2^2 and one Radix-2^3 modules (2^3-2^2-2^3), as it can be carried out with the sequence 2^3-2^3-2^2.

The proposed architecture implements the FFT algorithm, for sizes in correspondence with $N = 2^{n_S}$ ranging from 16 through 1024 with various combinations. The designs were evaluated in FPGA resource usage, performance and dynamic power consumption. FPGA Block Random Access Memory (BRAM) hardware was used in the scheduling unit and ROM operations. The Element delays implemented in shuffling and data synchronization operations were implemented with Shift Register Look-up tables (SRL). Mathematical DSP48E structures were used for standard multiplication, while all other design elements were synthesized with FPGA logic resources. Data Word Length (DWL) and Twiddle Factor Word Length (TFWL) were configured for 16 and 12 bits.

The FPGA utilized resources for the proposed high-frequency architecture are presented in Tables II and III in comparison with equivalent low-frequency architectures. Note low-frequency architectures refer to architectures without any pipeline registers or additional delay elements. A Resource Utilization Ratio (RUR) is utilized to provide a fair comparison, where the RUR compares the number of FPGA slices utilized in the proposed architecture R_p against the number of slices involved in an equivalent low-frequency design R_e; $RUR = \frac{R_p}{R_e}$.

Both high-frequency and low-frequency architecture designs obtained the same number of BRAM and DSP48E structures. Additionally, the RUR produced values did not exceed 1.74 with significantly high-level of throughput. Furthermore, building a FFT process with various Radix-2^2 and Radix-2^3 combinations offers significant advantages, including the reduction of FPGA resources and flexibility to FPGA platform resources. For example, a FFT size *N=256* denoted with the architecture structure 2^2-2^2-2^2-2^2 requires 42 BRAM and 36 DSP48E structures, while in comparison a 2^3-2^2-2^3 structure requires 38 BRAM and 28 DSP48E structures, while the 2^3-2^3-2^2 combination requires 40 BRAM and 32 DSP48E structures.

It is important to note the FPGA's occupied area average, which is calculated with averaging the percentage of used BRAMs, Slices and DSP48E blocks on the architecture implementation. The obtained FPGA occupied area average metrics range from 0.82% through 2.88 % within two parallel designs and from 1.89% through 4.96% within four parallel circuits. This demonstrates the proposed FFT process occupies a very small portion of the total area on a FPGA device, where the remaining area could be used to perform other MIMO-OFDM related functions.

TABLE II. VIRTEX-5 FPGA XC5VSX240T-2FF1738 RESOURCE USAGE OF THE PROPOSED MR2^2-R2^3 MDC FOR TWO PARALLEL DATA STREAMS

N	Structure	Occupied Slices			BRAM	DSP48E
		Low-Frequency	High-Frequency	RUR		
16	2^2-2^2	279	347	1.24	8	8
32	2^3-2^2	400	501	1.25	8	8
	2^2-2^3	391	535	1.37	8	8
64	2^2-2^2-2^2	413	534	1.29	12	16
	2^3-2^3	553	674	1.22	8	8
128	2^2-2^2-2^3	537	684	1.27	12	16
	2^3-2^2-2^2	557	679	1.22	12	16
256	2^2-2^2-2^2-2^2	685	760	**1.11**	16	24
	2^3-2^2-2^3	709	906	1.28	12	16
	2^3-2^3-2^2	703	851	1.21	12	16
512	2^3-2^3-2^3	1012	1280	1.26	12	16
	2^3-2^2-2^2-2^2	937	1141	1.22	16	24
	2^2-2^2-2^2-2^3	781	1102	**1.41**	16	24
1024	2^2-2^2-2^2-2^2-2^2	1089	1372	1.26	20	32
	2^3-2^3-2^2-2^2	1171	1386	1.18	16	24
	2^3-2^2-2^2-2^3	1247	1655	1.33	16	24

TABLE III. VIRTEX-5 FPGA XC5VSX240T-2FF1738 RESOURCES USAGE OF THE PROPOSED MR2^2-2^3 MDC FOR FOUR PARALLEL DATA STREAMS

N	Structure	Occupied Slices			BRAM	DSP48E
		Low-Frequency	High-Frequency	RUR		
16	2^2-2^2	508	619	1.22	30	12
32	2^3-2^2	693	991	1.43	32	16
	2^2-2^3	751	931	1.24	30	12
64	2^2-2^2-2^2	633	895	1.41	36	24
	2^3-2^3	924	1284	1.39	32	16
128	2^2-2^2-2^3	992	1123	1.13	36	24
	2^3-2^2-2^2	935	1262	1.35	38	28
256	2^2-2^2-2^2-2^2	629	1095	**1.74**	42	36
	2^3-2^2-2^3	1256	1744	1.39	38	28
	2^3-2^3-2^2	1222	1518	1.24	40	32
512	2^3-2^3-2^3	1656	2067	1.25	40	32
	2^3-2^2-2^2-2^2	1378	1596	1.16	44	40
	2^2-2^2-2^2-2^3	1607	1554	0.97	42	36
1024	2^2-2^2-2^2-2^2-2^2	1850	1659	**0.90**	48	48
	2^3-2^3-2^2-2^2	2063	2342	1.14	46	44
	2^3-2^2-2^2-2^3	1897	2433	1.28	44	40

The performances of the proposed high-frequency designs are compared with equivalent low-frequency designs in Table IV and Table V, demonstrating the system throughput in Mega-symbols per Second (Ms/S) and latency values. The proposed system comparison utilizes an Area to Throughput Ratio (ATR), calculated by dividing the number of occupied slices with the achieved throughput. The proposed architecture demonstrates significant throughput metric values ranging from 858 Ms/S to 916 Ms/S in two stream systems and from 1600 Ms/S to 1756 Ms/S in four stream systems. In comparison, the equivalent low-frequency architecture performance metrics range from 41.4 Ms/S to 125.6 Ms/S for two stream and from 86 Ms/S to 250.4 Ms/S in four stream designs.

The significant throughput improvement is not penalized with excessive resource usage and is efficient with device resource usage and achievable throughput as demonstrated with ATR values. Furthermore, the high-frequency architecture demonstrated significantly lower latency time than equivalent low-frequency architectures.

TABLE IV. VIRTEX-5 FPGA XC5VSX240T-2FF1738 PERFORMANCES OF THE PROPOSED MR2^2-2^3 FOR TWO PARALLEL DATA STREAMS

N	Structure	Low-Frequency			High-Frequency		
		Through-put(Ms/S)	ATR	Latency (nS)	Through-put(Ms/S)	ATR	Latency (nS)
16	2^2-2^2	**125.6**	**2.22**	127	**916**	**0.38**	48
32	2^3-2^2	91.0	4.40	352	910	0.55	81
	2^2-2^3	125.2	3.12	256	894	0.60	85
64	2^2-2^2-2^2	83.4	4.95	767	888	0.60	126
	2^3-2^3	87.0	6.36	736	878	0.77	139
128	2^2-2^2-2^3	80.0	6.71	1600	884	0.77	217
	2^3-2^2-2^2	60.6	9.19	2112	878	0.77	216
256	2^2-2^2-2^2-2^2	60.6	11.30	4224	884	0.86	367
	2^3-2^2-2^3	60.6	11.70	4224	872	1.04	383
	2^3-2^3-2^2	50.0	14.06	5120	880	0.97	377
512	2^3-2^3-2^3	50.0	20.24	10240	890	1.44	679
	2^3-2^2-2^2-2^2	48.8	19.20	10492	880	1.30	675
	2^2-2^2-2^2-2^3	58.8	13.28	8707	880	1.25	677
1024	2^2-2^2-2^2-2^2-2^2	47.6	22.88	21513	**858**	1.60	1296
	2^3-2^3-2^2-2^2	**41.4**	**28.29**	24734	862	1.61	1299
	2^3-2^2-2^2-2^3	47.8	26.09	21423	866	**1.91**	1293

TABLE V. VIRTEX-5 FPGA XC5VSX240T-2FF1738 PERFORMANCES OF THE PROPOSED MR2^2-2^3 FOR FOUR PARALLEL DATA STREAMS

N	Structure	Low Frequency			High-Frequency		
		Through-put(Ms/S)	ATR	Latency (nS)	Through-put(Ms/S)	ATR	Latency (nS)
16	2^2-2^2	**250.4**	**2.03**	64	**1756**	**0.35**	36
32	2^3-2^2	186.4	3.72	172	1696	0.58	64
	2^2-2^3	225.6	3.33	142	1688	0.55	64
64	2^2-2^2-2^2	222.8	2.84	287	1684	0.53	88
	2^3-2^3	174.4	5.30	367	1668	0.77	101
128	2^2-2^2-2^3	200.4	4.95	639	1668	0.67	144
	2^3-2^2-2^2	133.6	7.00	958	1644	0.77	146
256	2^2-2^2-2^2-2^2	195.6	3.22	1309	1616	0.68	233
	2^3-2^2-2^3	133.6	9.40	1916	1652	1.06	240
	2^3-2^3-2^2	105.2	11.62	2433	1668	0.91	237
512	2^3-2^3-2^3	100.0	16.56	5120	1636	1.26	416
	2^3-2^2-2^2-2^2	117.6	11.72	4354	1648	0.97	400
	2^2-2^2-2^2-2^3	190.8	8.42	2683	1668	0.93	396
1024	2^2-2^2-2^2-2^2-2^2	170.4	10.86	6009	**1600**	1.04	738
	2^3-2^3-2^2-2^2	**86.0**	**23.99**	11907	1644	1.42	730
	2^3-2^2-2^2-2^3	118.0	16.08	8678	1620	**1.50**	741

The dynamic power performance of the proposed high-frequency MR-2^2-2^3 MDC architecture is compared with low-frequency equivalent architecture results and presented in Table VI and Table VII. High-frequency architecture demonstrates a significant increase of dynamic power consumption compared to low-frequency architecture. However, the Dynamic Power to Throughput Ratio (DPTR) for the proposed architecture is lower than their equivalent low-frequency architecture for FFT size above 64. This demonstrates there is no additional power consumption when using one high-frequency circuit opposed to several low-frequency circuits to achieve the same throughput. Additionally the four data stream high-frequency architecture obtained an average power increase of approximately 100% in comparison to the equivalent two data stream architectures. The dynamic power consumption

TABLE VI. DYNAMIC POWER CONSUMPTION (W) OF THE PROPOSED TWO PARALLEL MR-2^2-2^3 MDC ARCHITECTURE IMPLEMENTED ON VIRTEX-5 FPGA XC5VSX240T-2FF1738

N	Structure	Low-Frequency		High-Frequency	
		Dynamic Power (W)	DPTR	Dynamic Power(W)	DPTR
16	2^2-2^2	0.128	1.02	1.079	1.18
32	2^3-2^2	0.110	1.21	1.095	1.20
	2^2-2^3	0.147	1.17	1.145	1.28
64	2^2-2^2-2^2	0.107	1.28	1.162	1.31
	2^3-2^3	0.111	1.28	1.176	1.34
128	2^2-2^2-2^3	0.106	1.33	1.180	1.33
	2^3-2^2-2^2	0.087	**1.44**	1.186	**1.35**
256	2^2-2^2-2^2-2^2	0.091	**1.50**	1.101	**1.25**
	2^3-2^2-2^3	0.092	**1.52**	1.044	**1.20**
	2^3-2^3-2^2	0.077	**1.54**	0.968	**1.10**
512	2^3-2^3-2^3	0.089	**1.78**	1.120	**1.26**
	2^3-2^2-2^2-2^2	0.087	**1.78**	1.190	**1.35**
	2^2-2^2-2^2-2^3	0.095	**1.62**	1.227	**1.39**
1024	2^2-2^2-2^2-2^2-2^2	0.092	**1.93**	1.345	**1.57**
	2^3-2^3-2^2-2^2	0.084	**2.03**	1.291	**1.50**
	2^3-2^2-2^2-2^3	0.088	**1.84**	1.364	**1.58**

TABLE VII. DYNAMIC POWER CONSUMPTION (W) OF THE PROPOSED FOUR PARALLEL MR-2^2-2^3 MDC ARCHITECTURE IMPLEMENTED ON VIRTEX-5 FPGA XC5VSX240T-2FF1738

N	Structure	Low Frequency		High-Frequency	
		Dynamic Power (W)	DPTR	Dynamic Power (W)	DPTR
16	2^2-2^2	0.295	1.18	2.16	1.23
32	2^3-2^2	0.236	1.27	2.251	1.33
	2^2-2^3	0.253	1.12	2.225	1.32
64	2^2-2^2-2^2	0.278	1.25	2.239	1.33
	2^3-2^3	0.225	1.29	2.346	1.41
128	2^2-2^2-2^3	0.264	**1.32**	2.033	**1.22**
	2^3-2^2-2^2	0.183	**1.37**	1.817	**1.11**
256	2^2-2^2-2^2-2^2	0.261	**1.33**	1.897	**1.17**
	2^3-2^2-2^3	0.195	**1.46**	2.069	**1.25**
	2^3-2^3-2^2	0.159	**1.51**	1.811	**1.09**
512	2^3-2^3-2^3	0.165	**1.65**	2.107	**1.29**
	2^3-2^2-2^2-2^2	0.182	**1.55**	2.071	**1.26**
	2^2-2^2-2^2-2^3	0.284	**1.49**	2.191	**1.31**
1024	2^2-2^2-2^2-2^2-2^2	0.270	**1.58**	2.132	**1.33**
	2^3-2^3-2^2-2^2	0.160	**1.86**	1.708	**1.04**
	2^3-2^2-2^2-2^3	0.201	**1.70**	2.113	**1.30**

demonstrates relatively little difference for large FFT size *N* with different Radix-2^2 and Radix-2^3 module combinations.

The proposed high-frequency architecture was carried out on a Virtex-7 XC7VS485T-2FFG1661 FPGA device to produce more recent device results obtained for FPGA area resources, maximum frequency and dynamic power consumption. The proposed designs FPGA area resource utilization is presented in Table VIII. The Virtex-7 implementation demonstrated significant savings in FPGA occupied slices, producing up to 28% and 23% slice reduction in comparison to Virtex-5 for two and four data stream implementations, which are presented in Tables II and Table III. Note the Virtex-7 FPGA's average occupied area is not exceeding 1.17% and 2.11% within two-parallel and four-parallel designs.

TABLE VIII. VIRTEX-7 FPGA XC7VS485T-2FFG1661 RESOURCES USAGE OF THE PROPOSED MR2^2-2^3

N	Structure	Occupied Slices	
		Two parallel	Four parallel
16	2^2-2^2	278	606
32	2^3-2^2	391	850
	2^2-2^3	413	882
64	2^2-2^2-2^2	388	814
	2^3-2^3	539	1141
128	2^2-2^2-2^3	556	1059
	2^3-2^2-2^2	529	1054
256	2^2-2^2-2^2-2^2	587	1033
	2^3-2^2-2^3	722	1352
	2^3-2^3-2^2	706	1367
512	2^3-2^3-2^3	958	1694
	2^3-2^2-2^2-2^2	828	1404
	2^2-2^2-2^2-2^3	858	1405
1024	2^2-2^2-2^2-2^2-2^2	1052	1569
	2^3-2^3-2^2-2^2	1182	1920
	2^3-2^2-2^2-2^3	1187	1872

The maximum frequency for the proposed architectures synthesized on Virtex-7 and Virtex-5 platforms over increasing FFT size are presented in Fig. 12. Virtex-7 implemented architectures achieved higher frequency values in comparison to Virtex-5. Virtex-7 implementations initially achieved 477 MHz followed with very low-levels of decreasing frequency as FFT size increases. Virtex-5 implementations achieved lower frequency with 458 MHz and 439 MHz for two and four data stream architecture with relatively large frequency decrease as FFT size increases.

As shown in Fig. 13 the dynamic power consumption of the proposed architecture designs implemented on Virtex-7 demonstrated significant power savings in comparison to Virtex-5 implementations. For FFT size *N=16* architecture at 400 MHz maximum frequency, two data stream demonstrated initial dynamic power consumption of 0.9 W and 0.4 W for Virtex-5 and Virtex-7 implementations, while four data streams obtained 2 W and 0.9 W for Virtex-5 and Virtex-7 implementations. It is important to note all designs demonstrated very low-levels of dynamic power consumption increase as FFT size increases.

The dynamic power consumption of high-frequency architecture for FFT size *N=16* and 1024 with increasing frequency, synthesized for Virtex-5 and Virtex-7 platforms are presented in Fig. 14a and Fig. 14b. Virtex-7 implementations demonstrate significant dynamic power reductions in comparison to Virtex-5, where the dynamic power reduction magnitude increases as frequency increases.

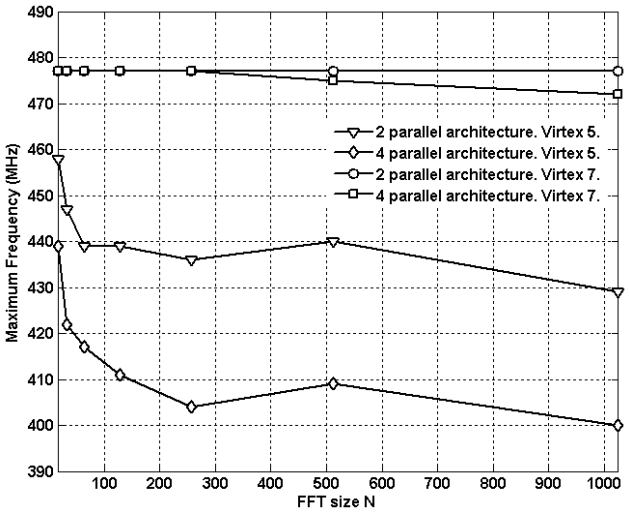


Figure 12. Maximum frequency versus FFT size *N* comparison between Virtex-7 and Virtex-5 implementation of proposed architectures

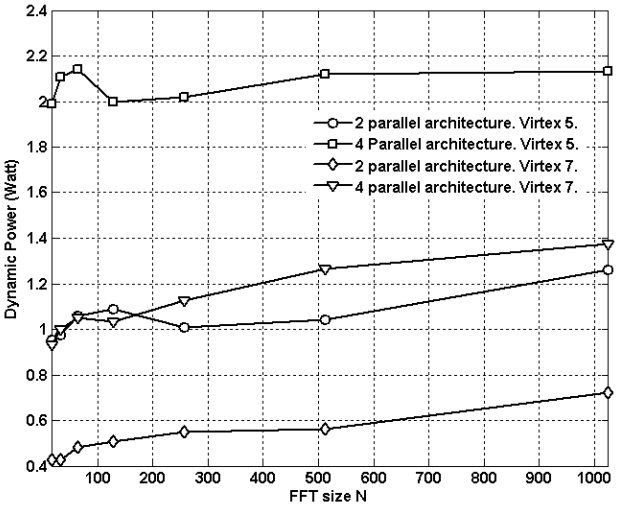
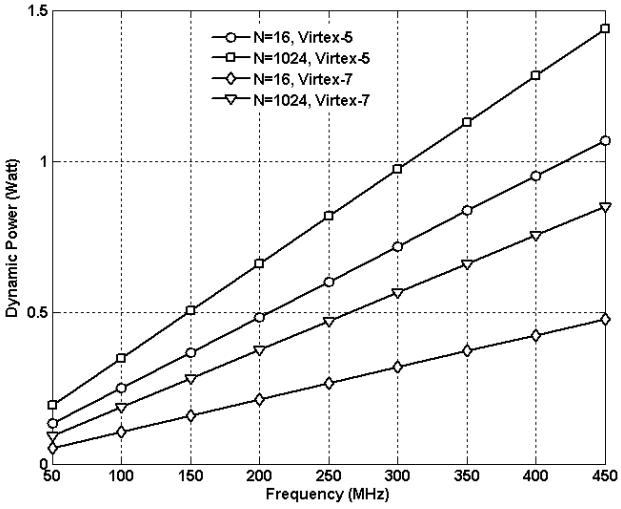
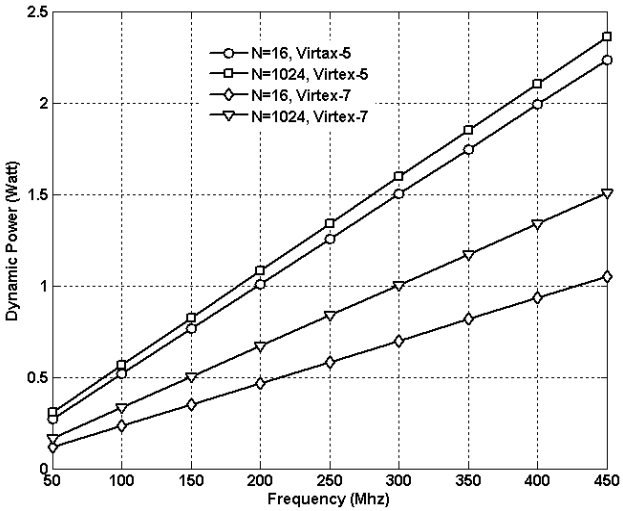


Figure 13. Dynamic power versus FFT size *N* comparison between Virtex-7 and Virtex-5 implementation of proposed architectures (with *F=400 MHz*)



(a)



(b)

Figure 14. Dynamic power versus frequency comparison between Virtex-7 and Virtex-5 implementation of proposed architectures. (a) Two parallel architecture. (b) Four parallel architecture

The proposed high-frequency architecture is compared with recent and comparable techniques within literature as presented in Table IX. It is important to note Ayinala, Brown and Parhi [20] introduced a two-parallel architectures for one data stream without an input scheduling unit, while Wang, Liu, He and Yu [23] presented a single path FFT architecture to obtain a DFT response for a single stream. In order to provide a valid comparison the compared resources have been scaled to reflect the articles presented number of parallel paths. The proposed high-frequency design demonstrated the highest achieved frequency amongst the implementation techniques compared, where the proposed method obtains a significant advantage. The proposed architectures frequency improvement when compared against equivalent techniques increases as FFT size N increases.

TABLE IX. COMPARISON OF THE PROPOSED ARCHITECTURES FPGA IMPLEMENTATION WITH PREVIOUS WORKS

N	Method	Device	Slice LUTs	Slices	DSPs	Frequency (MHz)
16	**Proposed**	**XC5VSX240T-2FF1738**	**1355**	**619**	**12**	**439**
		XC7VS485T-2FFG1661	**1575**	**606**	**12**	**477**
		XC6CSX475T-1 FF1156	**1307**	**506**	**12**	**400**
		XC5VLX20T	**1136**	**534**	**12**	**384**
	Garrido *et al.* [21]	XC5VSX240T-2FF1738	-	386	12	458
	Garrido *et al.* [22]	XC6CSX475T-1 FF1156	-	567	12	335
	Ayinala *et al.* [20]	XC5VLX20T	2348	1046	-	370
	Wang *et al.* [23]	XC5VSX240T-2FF1738	2688		16	322
32	**Proposed**	**XC5VSX240T-2FF1738**	**2301 - 2327**	**931 - 991**	**12 - 16**	**422 - 424**
		XC7VS485T-2FFG1661	**2527 - 2534**	**850 - 882**	**12 - 16**	**471 - 477**
		XC5VLX20T	**2109 - 2135**	**829 - 835**	**12 - 16**	**382 - 385**
	Ayinala *et al.* [20]	XC5VLX20T	3088	1290	-	370
64	**Proposed**	**XC5VSX240T-2FF1738**	**2059 - 3294**	**895 - 1284**	**16 - 24**	**417 - 421**
		XC7VS485T-2FFG1661	**2258 - 3466**	**814 - 1141**	**16 - 24**	**475 - 477**
		XC6CSX475T-1 FF1156	**2831 - 4028**	**699 - 1006**	**16 - 24**	**394 - 400**
		XC5VLX20T	**2007 - 3026**	**811 - 1151**	**16 - 24**	**375 - 394**
	Garrido *et al.* [21]	XC5VSX240T-2FF1738	-	695	24	384
	Garrido *et al.* [22]	XC6CSX475T-1 FF1156	-	782	24	335
	Ayinala *et al.* [20]	XC5VLX20T	3832	1560	-	370
	Wang *et al.* [23]	XC5VSX240T-2FF1738	4440	-	32	303
256	**Proposed**	**XC5VSX240T-2FF1738**	**2914 - 4152**	**1095 - 1744**	**28 - 36**	**404 - 417**
		XC7VS485T-2FFG1661	**3117 - 4334**	**1033 - 1367**	**28 - 36**	**472 - 477**
		XC6CSX475T-1 FF1156	**2837-4046**	**928 - 1305**	**28 - 36**	**400**
	Garrido *et al.* [21]	XC5VSX240T-2FF1738	-	1024	36	389
	Garrido *et al.* [22]	XC6CSX475T-1 FF1156	-	924	36	240
	Wang *et al.* [23]	XC5VSX240T-2FF1738	6932	-	48	297
1024	**Proposed**	**XC5VSX240T-2FF1738**	**4932 - 5483**	**1659 - 2433**	**40 - 48**	**400 - 411**
		XC7VS485T-2FFG1661	**5063 - 6307**	**1569 - 1920**	**40 - 48**	**472 - 474**
		XC6CSX475T-1 FF1156	**4828-6011**	**1434 - 1824**		**400**
	Garrido *et al.* [21]	XC5VSX240T-2FF1738	-	1425	48	270
	Garrido *et al.* [22]	XC6CSX475T-1 FF1156	-	1351	48	227
	Wang *et al.* [23]	XC5VSX240T-2FF1738	11216	-	64	298

The proposed realization involves fewer DSP mathematical hardware operators than techniques demonstrated in [23] and fewer or the same number of DSP operators than references [21, 22] for all FFT sizes investigated. Additionally, the presented architecture required fewer slices Look-Up Tables (LUT) than references [20] and [23] and fewer occupied slices than [20]. It is challenging to provide a valid occupied slice resource comparison with [21] and [22] as the compared techniques do not include an input shuffling unit function, which is necessary to process multiple streams in parallel.

VI. CONCLUSION

This article presents an efficient FPGA implementation of high-frequency FFT architecture based on mixed Radix-2^2 Radix-2^3 MDC structures. The presented architecture design can process two or four independent data streams in parallel with efficient utilization of FPGA resources and demonstrates a significant high-throughput performance in comparison to other equivalent techniques. The designs were intended for MIMO-OFDM communication systems, where parallel data processing is a critical process, requiring efficient resource usage and high-throughput. The system is very simple to control with binary counter signals. The presented architecture is based on mixed Radix structures to realize scalable architectures to any FFT size $N = 2^{n_s}$. Additionally, the mixed Radix approach offers advantages in the implementation flexibility, where a FFT processor for each size can be realized with several differing Radix-2^2 and Radix-2^3 module combinations. The designs were carried out and evaluated on Virtex-5 and Virtex-7 FPGA devices. Experimental results demonstrate the presented high-frequency architectures obtained throughput ratio improvements of 20.82 within two-parallel and 19.12 within

four-parallel architectures in comparison to equivalent low-frequency designs. The proposed architecture obtained maximum frequency values of 458 MHz and 477 MHz for Virtex-5 and Virtex-7 devices, in addition to obtaining very low-levels of performance decrease as FFT size *N* increases. Additionally, Virtex-7 device implementations demonstrated significant dynamic power reduction while obtaining significantly higher throughput than Virtex-5.

REFERENCES

[1] D. Gesbert, M. Shafi, D. Shiu, P.J. Smith, and A. Naguib, "From theory to practice: an overview of MIMO space–time coded wireless systems," IEEE J. Select. Areas Commun., vol. 21, no. 3, pp. 281–302, Apr 2003. doi: 10.1109/JSAC.2003.809458.

[2] J. A. C. Bingham, "Multicarrier modulation for data transmission: an idea whose time has come," IEEE Communications Magazine, vol. 28, pp. 5–14, May 1990. doi: 10.1109/35.54342.

[3] H. Sampath, S. Talwar, J. Tellado, V. Erceg, A. Paulraj, "A fourth generation MIMO-OFDM: broadband wireless system: Design, performance, and field trial results," Communications Magazine, IEEE, vol. 40, no. 9, pp. 143-149, Sep. 2002. doi: 10.1109/MCOM.2002.1031841.

[4] Y. G. Li, J. H. Winters, N. R. Sollenberger, "MIMO-OFDM for wireless communications: Signal detection with enhanced channel estimation," IEEE Trans. Communications, vol. 50, no. 9, pp. 1471-1477, Sep. 2002. doi: 10.1109/TCOMM.2002.802566.

[5] H. Y. Chen, J. N. Lin, H. S. Hu, S. J. Jou, "STBC-OFDM downlink baseband receiver for mobile WMAN," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 21,no. 1, pp. 43-54, Jan 2013. doi: 10.1109/TVLSI.2011.2181965.

[6] IEEE 802.16 Working Group, "IEEE standard for local and metropolitan area networks. Part 16: Air interface for fixed broadband wireless access systems," IEEE Standard 802.16e-2005, 2006. doi: 10.1109/IEEESTD.2006.99107.

[7] Y. W. Lin, C. Y. Lee, "Design of an FFT/IFFT processor for MIMO OFDM systems," IEEE Trans. on Circuits and Systems I, vol. 54, no. 4, pp. 807-815, Apr. 2007. doi: 10.1109/TCSI.2006.888664.

[8] B. Fu, P. Ampadu, "An area efficient FFT/IFFT processor for MIMO OFDM WLAN 802.11n," Journal of Signal Processing Systems, Springer, vol. 56, no. 1, pp. 59-68, Jul. 2009. doi:10.1007/s11265-008-0264-9.

[9] K. J. Yang, S. H. Tsai, , G. C. H. Chuang, "MDC FFT/IFFT Processor With Variable Length for MIMO-OFDM Systems," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 4 , pp. 720-731, Apr. 2013. doi: 10.1109/TVLSI.2012.2194315.

[10] S. N. Tang, C. H. , Liao, T. Y. Chang, "An area- and energy-efficient multimedia FFT processor for WPAN/WLAN/WMAN systems," IEEE J. of Solid-State Circuits, vol. 47, no. 6, pp. 1419-1435, Jun. 2012. doi: 10.1109/JSSC.2012.2187406.

[11] S. N. Tang, J. W. Tsai, T. Y. Chang, "A 2.4 GS/s FFT processor for OFDM based WPAN applications," IEEE Trans. on Circuits and Systems II: Express Briefs, vol. 57, no. 6, pp. 451-455, Jun. 2010. doi: 10.1109/TCSII.2010.2048373.

[12] C. Wang, Y. Yan, X. Fu, "A High-Throughput Low-Complexity Radix-24-2^2-2^3 FFT/IFFT Processor With Parallel and Normal Input/Output Order for IEEE 802.11ad Systems," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 11, pp. 2728-2732, Nov. 2015. doi: 10.1109/TVLSI.2014.2365586.

[13] P. Y. Tsai, C. W. Chen, M. Y. Huang, "Automatic IP generation of FFT/IFFT processors with word-length optimization for MIMO-OFDM systems," EURASIP J. of Advances in Signal Processing, vol. 2011, no. 1, pp. 1-15, Jan. 2011. doi: 10.1155/2011/136319.

[14] Y. W. Lin, H. Y. Liu, C. Y. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications," IEEE J. of Solid-State Circuits, 2005, vol. 40, no. 8, pp. 1726-1735, Aug. 2005. doi: 10.1109/JSSC.2005.852007.

[15] Y. T. Lin, P. Y. Tsai, T. D. Chiueh, "Low-power variable-length fast Fourier transform processor," in IEE Proc. Computers and Digital Techniques, vol. 152, no. 4, pp. 499-506, Jul. 2005. doi: 10.1049/ip-cdt:20041224.

[16] S. He, M. Torkelson, "A new approach to pipeline FFT processor," in Proc. International Parallel Processing Symposium (IPPS '96), Washington, DC, Apr. 1996, pp. 766-770. doi: 10.1109/IPPS.1996.508145.

[17] S. He, M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in Proc. International Signals, Systems, and Electronics Symposium (ISSSE 98), Pisa, Sep. 1998, pp. 257- 262. doi: 10.1109/ISSSE.1998.738077.

[18] P. P. Boopal, M. Garrido, O. Gustafsson, "A reconfigurable FFT architecture for variable-length and multi-streaming OFDM standards," in Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2013), Beijing, May 2013, pp. 2066-2070. doi: 10.1109/ISCAS.2013.6572279.

[19] E. E. Swartzlander, W. K. W. Young, S. J. Joseph, "A radix-4 delay commutator for fast Fourier transform processor implementation," IEEE J. of Solid-State Circuits, vol. 19, no. 5, pp. 702-709, Oct. 1984. doi: 10.1109/JSSC.1984.1052211.

[20] M. Ayinala, M. Brown, K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 6, pp. 1068-1081, Jun. 2012. doi: 10.1109/TVLSI.2011.2147338.

[21] M. Garrido, J. G. Rajal, M. A.. Sánchez, O. Gustafsson, "Pipelined Radix-2k Feedforward FFT Architectures," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 1, pp. 23-32, Jan. 2013. doi: 10.1109/TVLSI.2011.2178275.

[22] M. Garrido, M. Acevedo, A.. Ehliar, O. Gustafsson, "Challenging the limits of FFT performance on FPGAs," in Proc. IEEE International Symposium on Integrated Circuits (ISIC), Singapore, Dec. 2014, pp. 172-175. doi: 10.1109/ISICIR.2014.7029571.

[23] Z. Wang, X. Liu, B. He, F. Yu, "A Combined SDC-SDF Architecture for Normal I/O Pipelined Radix-2 FFT," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 5, pp. 793-977, May 2015. doi: 10.1109/TVLSI.2014.2319335.

[24] S. Uzun, A. Amira, A. Bouridane, "FPGA implementations of fast Fourier transforms for real-time signal and image processing," in IET Proc. in Vision, Image and Signal Processing, vol. 152, no. 3, pp. 283-296, Jun. 2005. doi: 10.1049/ip-vis:20041114